

ORACLE®



ORACLE®

Third Party Software – Some Security Considerations

John Heimann

Vice President, Security Program Management

Global Product Security

Third Party Software

- Oracle products (and those of many other vendors) embed hundreds of third party libraries including
 - Commercially-licensed
 - Free and Open Source Software (FOSS)
- There are many reasons for using third party software
 - Reduced development and/or support cost
 - Time to market
 - Best of breed
 - Interoperability

Commercial Third Party Software

- Usually closed source (but may have open version)
- Typically requires payment of license and support fees
- Often have well established processes and schedules for:
 - Functional releases
 - Maintenance updates
 - Support
 - End of life

Commercial Third Party Software, cont.

- *May* have mature software security development lifecycle (SDLC)
- *May* have well-established:
 - Security patch schedule
 - Policies for security bug handling and vulnerability disclosure
- Can be single neck to choke in case of problems
 - Assuming reasonable support contract for security fixes, end of life dates, etc.

Oracle Software Security Assurance (OSSA)



- Defined and managed by Global Product Security (GPS)
- The heart of GPS' mission
- Evolved over two decades
 - New programs added as software assurance process and technology matures
- Common methodology adapted for each business and technology area

Free and Open Source Software (FOSS)

- Often developed through a FOSS community effort
- Does not require license fees or support fees
 - May have license terms with special considerations, such as GNU Public License (GPL)
 - Commercial support may be available for a fee (e.g., Oracle Linux)
- Development and release practices vary widely
 - May have predictable release or maintenance schedules
 - Rarely have well established end-of-life dates

FOSS, cont.

- Typically does not have formal SDLC
 - Software actively maintained by commercial companies, like Oracle Linux, Java, OracleVM and MySQL, are exceptions
- Security patches are usually ad hoc
- Patch quality is not uniform over time
- Security fixes may require upgrades to new versions with incompatible APIs
 - e.g., Struts1 to Struts2

Loose vs. Tight Coupling

- Commercial third party software typically delivered as object code with documented APIs
 - Typically loosely coupled to embedding product (e.g., at link time)
- FOSS allows tighter coupling of third party code and even modification of that code before it is embedded
 - May allow better performance, coordination, etc.
 - May allow customization of third party functionality
- Tighter coupling of third party code often means higher support cost

Some FOSS Myths

- Open source is free!
- “Million Eyes” means security

Open Source is Free!

- No license or support fees
- You get what you pay for, at least in support
 - No support service level agreements are available
 - Arbitrary release schedules for upgrades and fixes which may cause disruption to embedding product schedule
 - Organizations may need to change APIs to consume upgrades/fixes
 - Publicly known vulnerabilities aren't always fixed
- Review and approval for FOSS involves significant effort
- Security patch management for open source can be expensive

“Million Eyes” means Security

- Open source means that anyone could review the code
 - Including the “best minds in the world”
 - But do they?
- Reality
 - Code review is hard work, time consuming
 - Security tools require time, effort and money to run
 - Thousands of cursory reviews is not worth one careful review
 - Those who have time and money to find vulnerabilities in open source aren't always the good guys
 - Open source makes attacker's job easier
 - Vulnerabilities found by attackers have big impact if FOSS is widely used

Million Eyes Theory, cont.

- Unix BASH vulnerability (a.k.a., “Shellshock”)
 - BASH is widely used
 - Was present in BASH since version 1.0.3 (Sept. '89)
 - Not found until 25 years later (Sept. '14)
 - Very serious vulnerability – allows execution of arbitrary commands on unpatched servers
- Places serious doubt on effectiveness of “million eyes”
- Many other examples (e.g., OpenSSL, NSS)

Million Eyes Theory, cont.

- Customers often expect a patch each time there is a security patch release in a third party library
 - Tightly coupled third party code typically means Oracle has to issue a patch based on third party patch
- Security patch releases in 2014:
 - OpenSSL – 5 releases
 - NSS – 15 releases
 - Apache Tomcat – 7 releases
 - cURL – 6 releases
 - PHP – roughly one per month

Java Standard Edition (SE) is an interesting case study

- Prior to Oracle's acquisition of Sun, Java SE relied heavily on the "million eyes" of the community for security
 - Some vulnerabilities were found and fixed through the community process
 - Many more were found by professional researchers, and made public after Oracle's acquisition of Java
- Since Oracle's acquisition, Java has adopted Oracle security practices
 - Java development process has much greater pre-release security analysis and testing
 - Java security architecture has been improved
 - Better tools to manage Java security and remove old, vulnerable versions Java
 - *Significantly fewer vulnerabilities have been reported in Java*
- Java is still open source, but security improvements have come from Oracle's active investment in Java security

FOSS Vulnerabilities Can Have Severe Impact

- FOSS vulnerabilities can affect many vendors and end users, increasing
 - Public (and customer) awareness
 - Attention from hackers
 - Fear uncertainty and doubt in the media
 - Rush to fix, risk of poor fix quality
 - Fix schedule is unpredictable, cannot be coordinated with vendor or customer schedule

FOSS Vulnerabilities, cont.

- Publicity surrounding FOSS vulnerabilities increases pressure on Oracle and other vendors to issue fixes
 - May be disproportionate to actual severity of bug
 - Customers often demand one-off patches
- Compared with patching in regularly scheduled security patch releases, one-off patches increase
 - Patching cost for vendor and customers
 - Customer pain (out of cycle urgent patching)
 - Risk of negative publicity for vendor

Choosing Third Party Software – Commercial or FOSS

- Development teams who choose to embed third party software must consider the security lifecycle cost and risk.
- Oracle has a centralized, corporate-level third party software approval system since poor choice of third party software can lead to risk:
 - License
 - Competitive
 - Security

Before Choosing Commercial or FOSS Third Party Software – Get Security Relevant Information

- Determine what if any Secure Development Lifecycle was used in software development
- Review vulnerability and security patch history
- Determine what the support process, support life, and security patching process may be

Assess Software Secure Development Process

- Commercial software vendors may have mature secure development processes (but should be reviewed)
- Some FOSS also has this
 - Commercial software may include FOSS
- Oracle's OSSA or Microsoft's SDLC are good benchmark processes
- A mature secure development process helps prevent (but does not guarantee absence of) security vulnerabilities

Check Publicly Reported Vulnerabilities

- National Vulnerability Database is a good source of information
 - <https://nvd.nist.gov/>
 - Important to use correct software name and version
 - Assumes some knowledge of Common Weakness and Enumeration (CWE), Common Vulnerability Scoring System (CVSS)
- Software with many past vulnerabilities may be suspect
 - May reflect poor design or development process
 - May reflect unusual hacker attention
 - Note that commercial software may do “silent” bug fixing

Review Security Patch History

- Frequent security patches for software may complicate development and maintenance of products which embed that software
- Security fixes which require API changes make adoption of those fixes difficult
 - A history of these should raise concern

Ensure Support Path

- Determine if software is supported in an acceptable manner and if end-of-life is specified
 - Library should have consistent, multi-year history of security fixes
- Third party software must have support path through life of the embedding product
 - Do not choose third party software already past end-of-life
 - May require your product development team to support third party software
- Ensure upgrade path is available if third party end-of-life falls within support life of embedding product

Before Releasing Third Party Software in Your Products or Systems

- Get approvals, keep records
- Treat third party code like it's your code

Get Approvals, Keep Records

- Third party software use can introduce risk to your organization
 - A formal approval process for using third party software is recommended
 - Sign off from internal experts (corporate architecture, security, legal, etc.) where relevant
- Keeping record of third party software use is strongly recommended
 - Which specific third party libraries are in which specific product releases
 - Important for determining which products a third party vulnerability affects and/or where third party patches or upgrades must be applied

Treat Third Party Code Like It's Your Code

- Customers/users are affected by vulnerabilities in your software whether they are in your code or embedded third party code
 - Whose “fault” it may be is irrelevant if customer/user is hacked
 - Burden of due security diligence is on you
- If security analysis and testing of embedded third party product is possible, do it!
 - FOSS (or commercial code if you get source) can be subjected to static analysis
 - Open or closed source software can be subject to dynamic analysis tools, fuzzing, architectural risk analysis, etc.

Third Party Security Patches

- Depending on how tightly coupled a third party library is to software you develop which embeds that library, a third party security patch may
 - Require you to prepare a security patch based on third party patch and distribute it as a security patch to your software's customers/users (tightly coupled)
 - Allow your product's customers/users to download patch directly from third party source and install it themselves (loosely coupled)

Hardware and Software

ORACLE®

Engineered to Work Together